Embedded Vision Intelligent Laboratory

多媒體技術與應用
Spring 2021

Instructor： Yen-Lin Chen(陳彥霖), Ph.D.
Professor
Dept. Computer Science and Information Engineering
National Taipei University of Technology

Lecture 11

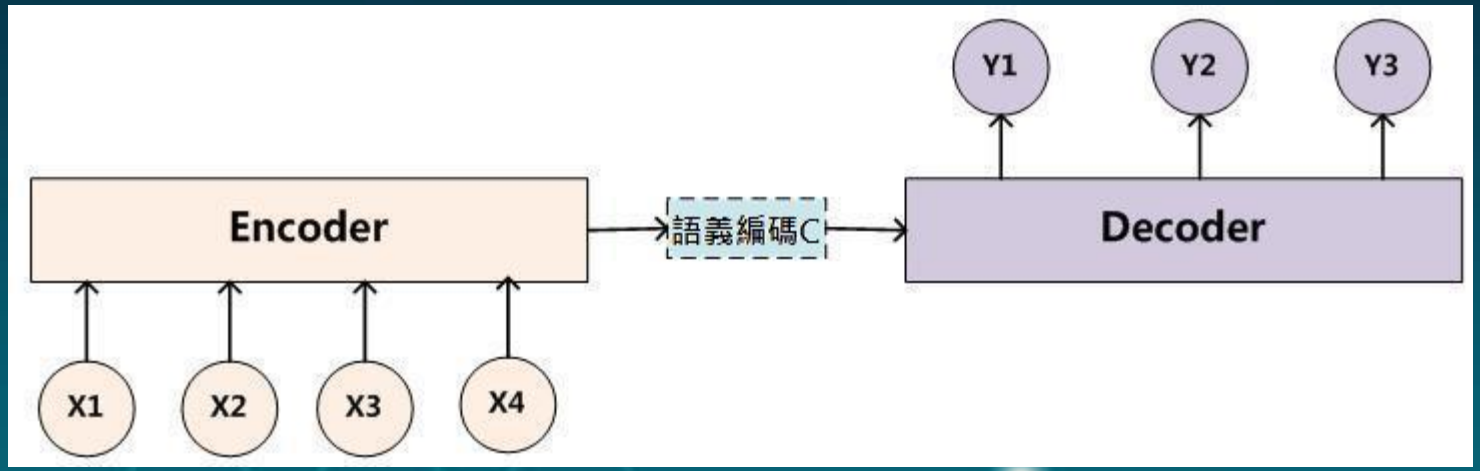
Social LSTM的發展與應用

Encoder-Decoder框架

- Encoder-Decoder（編碼-解碼）是深度學習中非常常見的一個模型框架，比如非監督演算法的auto-encoding就是用Encoder-Decoder的結構設計並訓練的；比如這兩年比較熱的image caption的應用，就是CNN-RNN的Encoder-Decoder框架；再比如神經網路機器翻譯NMT模型，往往就是LSTM-LSTM的Encoder-Decoder框架。因此，準確的說，**Encoder-Decoder並不是一個具體的模型，而是一類框架**。
- Encoder和Decoder部分可以是任意的文字，語音，影像，視訊資料，模型可以採用CNN，RNN，BiRNN、LSTM、GRU等等。所以基於Encoder-Decoder，我們可以設計出各種各樣的應用演算法。

Encoder-Decoder框架

- Encoder-Decoder框架可以看作是一種文字處理領域的研究模式，應用場景異常廣泛。下圖是文字處理領域裡常用的Encoder-Decoder框架最抽象的一種表示：



Encoder-Decoder框架

- Encoder-Decoder框架可以這麼直觀地去理解：可以把它看作適合處理由一個句子（或篇章）生成另外一個句子（或篇章）的通用處理模型。對於句子對<X,Y>，我們的目標是給定輸入句子X，期待通過Encoder-Decoder框架來生成目標句子Y。

Encoder-Decoder框架

- 而X和Y分別由各自的單字序列構成：

$$X = \langle x_1, x_2 \dots x_m \rangle$$
$$Y = \langle y_1, y_2 \dots y_n \rangle$$

- Encoder顧名思義就是對輸入句子X進行編碼，將輸入句子通過非線性變換轉化為中間語義表示C：

$$C = \mathcal{F}(x_1, x_2 \dots x_m)$$

- 對於解碼器Decoder來說，其任務是根據句子X的中間語義表示C和之前已經生成的歷史資訊$y_1, y_2 \dots y_{i-1}$來生成i時刻要生成的單字y_i。

$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

- 每個y_i都依次這麼產生，那麼看起來就是整個系統根據輸入句子X生成了目標句子Y。

Encoder-Decoder框架

- Encoder-Decoder不是一種模型，而是一種框架，一種處理問題的思路，最早應用於機器翻譯領域，輸入一個序列，輸出另外一個序列。機器翻譯問題就是將一種語言序列轉換成另外一種語言序列，將該技術擴充套件到其他領域，比如輸入序列可以是文字，語音，影像，視訊，輸出序列可以是文字，影像，可以解決很多別的型別的問題。這裡以輸入為文字，輸出也為文字作為例子進行介紹：

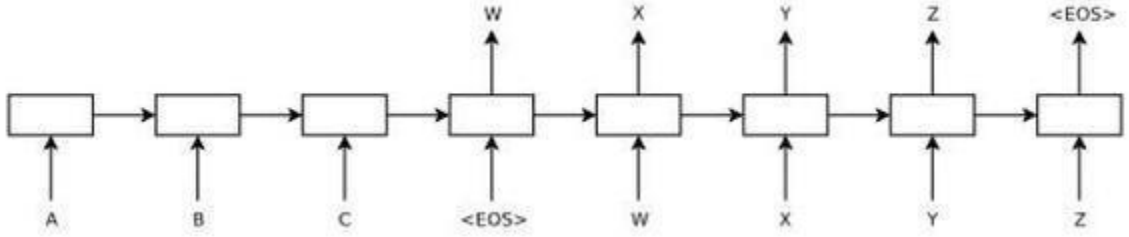


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Encoder-Decoder框架

- encoder部分是將輸入序列表示成一個帶有語義的向量，使用最廣泛的表示技術是Recurrent Neural Network，RNN是一個基本模型，在訓練的時候會遇到gradient explode或者gradient vanishing的問題，導致無法訓練，所以在實際中經常使用的是經過改良的LSTM RNN或者GRU RNN對輸入序列進行表示，更加複雜一點可以用BiRNN、BiRNN with LSTM、BiRNN with GRU、多層RNN等模型來表示，輸入序列最終表示為最後一個word的hidden state vector。

Encoder-Decoder框架

- decoder部分是以encoder生成的hidden state vector作為輸入“解碼”出目標文字序列，本質上是一個語言模型，最常見的是用Recurrent Neural Network Language Model (RNNLM)，只要涉及到RNN就會有訓練的問題，也就需要用LSTM、GRU和一些高階的model來代替。目標序列的生成和LM做句子生成的過程類似，只是說計算條件機率時需要考慮encoder向量。

Social LSTM

Social LSTM簡介

- 人類具有與生俱來的相互“閱讀(觀察)”的能力。當人們走在擁擠的公共場所，如人行道、機場航站大廈或購物中心時，他們會遵守大量(不成文的)常識規則，並遵守社會習俗。
- 例如：當他們考慮下一步移動到哪裡時，他們會尊重彼此的個人空間，並讓出通行權。

ref:https://zhuanlan.zhihu.com/p/342149300?fbclid=IwAR1L_eDWSSqe5pp5n7ip_8BjDlhFIbROlnv18W1w1TmPu509unIN4tkxpSY

Social LSTM簡介

- 對這些規則進行建模並使用它們來理解和預測複雜現實環境中的人體運動的能力對於廣泛的應用非常有價值：從**部署社會性機器人**到在**智慧環境中設計智慧跟蹤系統**等各類應用。然而，在考慮到這些常識行為的同時預測人類目標的運動是一個極具挑戰性的問題。在擁擠的空間裡，人們之間發生的複雜且往往微妙的互動。最近在電腦視覺領域的研究已經成功地解決了其中一些問題。Kitan等人已經證明，與忽略場景資訊的模型相比，關於靜態環境的語義(例如，人行道的位置、草地的延伸等)的推斷知識有助於更準確地預測未來場所中行人的軌跡。相關文獻的開創性工作還提出了對人與人之間的相互作用(通常稱為「社會力」)進行建模的方法，以提高多目標跟蹤問題的穩健性和準確性。

Social LSTM簡介

- 先前的工作大多受到以下兩個問題的限制。
  1. 使用手工提取的特徵來對特定設置的“交互”進行建模，而不是以資料驅動(**data driven**)的方式。這導致了只支持捕捉簡單交互(例如吸引/接觸)的模型，可能無法推廣到更複雜、擁擠的環境。
  2. 專注於對彼此接近的人之間的相互作用進行建模(以避免立即發生碰撞)。然而，它們並不能捕捉在更遙遠的未來可能發生的相互作用，缺乏長時依賴。

Social LSTM簡介

- 在這項工作中，作者提出了一種方法，可以通過一種新穎的資料驅動(**data driven**)架構來預測未來時刻的人類軌跡，從而解決這兩個挑戰。受長短期記憶網路(LSTM)在不同序列預測任務中的成功應用的啟發，也將其擴展到人體軌跡預測。雖然LSTM具有學習和複製長序列的能力，但它們不能捕獲多個相關序列之間的依賴關係。

Social LSTM簡介

- 論文通過一種新穎的架構來解決這個問題，該架構將對應於鄰近序列的LSTM連接起來。引入了一個 **Social池化層(Social-pooling)**，它允許空間中鄰近序列的LSTM彼此共用它們的隱藏狀態(ex.上週提過的單元狀態)。這種體系結構，稱之為 **Social-LSTM**，可以自動學習在時間上重合的軌跡之間發生的典型交互。這種模型利用現有的人類目標資料集，不需要任何額外的約束來學習人類在社交空間中觀察到的常識規則和慣例。

Social LSTM簡介

- 最後，在ETH和UCY這兩個公開可用的資料集上，證明提出的Social-LSTM能夠比當時最先進的方法更準確地預測行人的軌跡。除此之外，還分析了由該模型生成的軌跡模式，以理解從軌跡資料集中學習到的社會行為。

Social LSTM-方法

- 在擁擠的場景中移動的人會根據周圍其他人的行為來調整自己的動作。例如，一個人可以完全改變他/她的路線或暫時停下來，以容納一群向他移動的人。這種軌跡上的偏差不能通過孤立地觀察人來預測。也不能用簡單的“排斥”或“吸引”(傳統的社會力量模型)來解釋。

Social LSTM-方法

- 在這一部分中，可以參考圖一，這是一個基於**S-Pooling**的LSTM模型，這個模型預測場景中所有人的軌跡並進行交流。此方法被稱為Social LSTM模型。

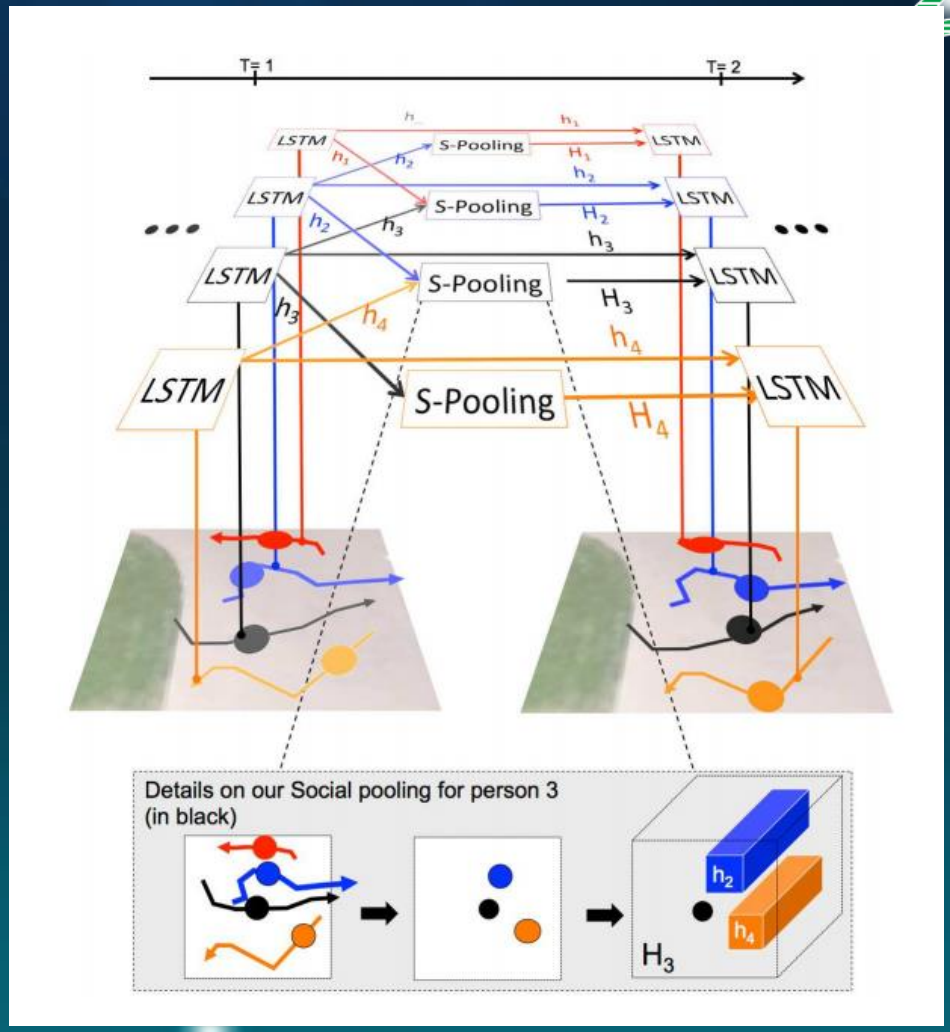
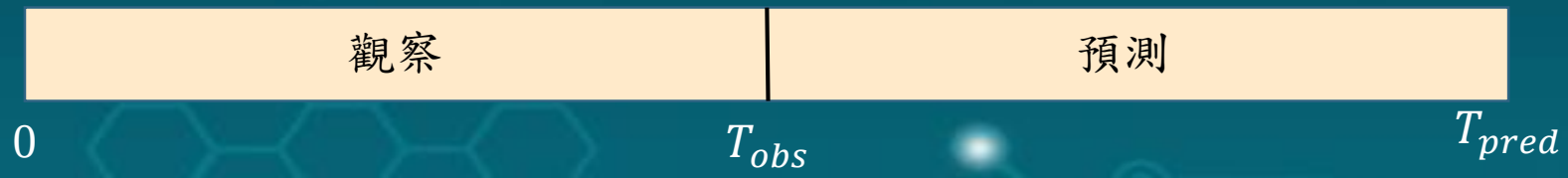


圖1. Social-LSTM方法整體流程圖，對於場景中每個個體使用單獨的LSTM網路。然後通過社交池化的方式相互連接，該池化層允許空間臨近的LSTM權重共用，通過這種方式網格範圍內的所有節點進行資訊融合

Social LSTM-方法

- 問題定義：
  - 假設首先對每個場景進行預處理，以獲得所有人在不同時刻的空間座標。以前的工作也遵循這一慣例。在任何時刻，場景中的人都由他/她的歐式座標 (x_t^i, y_t^i) 表示。我們觀察所有人從時間 $0 - T_{obs}$ 的位置，並預測他們在時間段 $T_{obs+1} - T_{pred}$ 的位置。這個任務也可以看作序列生成問題，其中輸入序列對應於行人的已觀察位置。



Social Pooling

- LSTM目前被證明應用於孤立的序列生成問題上效果較好，因此在相關論文中，對場景中每個行人都使用一個單獨的LSTM模型來進行其運動軌跡的學習。這種方式並沒有很好的捕捉鄰居節點之間的互動問題，單獨的LSTM對周圍節點的資訊不可知。為了結合環境中其他行人的特性，任意LSTM權重在一定範圍內的序列中是共用的，通過引入Social-Pooling(以下簡稱S-Pool)的方式來進行資訊的融合。

Social Pooling

- 首先解決一個問題：節點鄰居數量不同，這一問題在稠密網路中由於節點數量多尤為突出。因此首先考慮節點鄰居資訊的統一整合。在彙集資訊的同時，嘗試通過基於網格的彙集來保存空間資訊，如下頁所述：

Social Pooling

- LSTM在時間上的隱藏狀態捕捉到了此人在該時刻在場景中的最新表示。 我們通過建立一個“社交”隱藏狀態張量 H_t^i來與鄰居分享這種表示。在給定隱藏狀態維數D和鄰域大小 N_0的情況下，我們構造了其軌跡的在第i時刻的張量 $H_t^i \in R^{N_0 \times N_0 \times D}$ 。可以理解是以原先節點為中心圈定一個網格，對網格內的節點隱藏狀態進行池化操作，具體的方案是通過加法進行，這樣，周圍節點資訊均可通過一個 $R^{N_0 \times N_0 \times D}$ 大小的tensor來進行表示。

Social Pooling

- 具體公式如右：$H_t^i(,n,:) = \sum_{j \in N_i} 1mn\left[x_t^j - x_t^i, y_t^j - y_t^i\right] h_{t-1}^j$
- 其中 H_{t-1}^j 表示的是節點上一層的隱藏狀態; $1mn$ 是表示該節點是否在（m，n）網格中的指示器，在為1，不在為0; N_i 表示以 N_i 為中心所圈的網格內的所有節點，也就是所謂的鄰居節點。具體示意圖2如下：

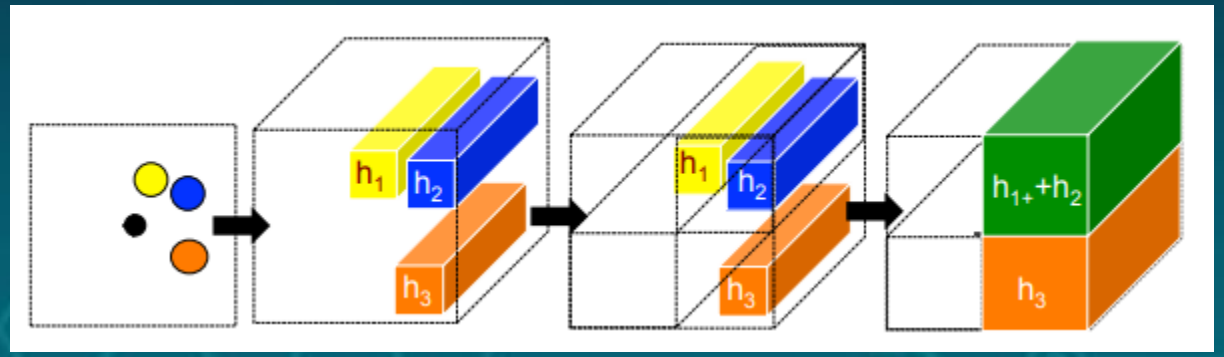


圖2. 上圖顯示了由黑點代表的人的社會池化。將鄰居的隱藏狀態(以黃色、藍色和橙色顯示)集中在一定的空間距離內。如最後兩個步驟所示，池中部分保留了鄰居的空間資訊，且以格式化的方式輸出

Social Pooling

- 如上頁圖中，以黑點為中心圈出一片區域，切分為網格，將網格內的隱藏狀態進行資訊聚合。分別使用全連接層將鄰居資訊成為tensor_embed和位置資訊input_embed通過串接(concat)加入再輸入下一時間的LSTM cell，一個時間步驟中的資料流程程大致如右圖所示：

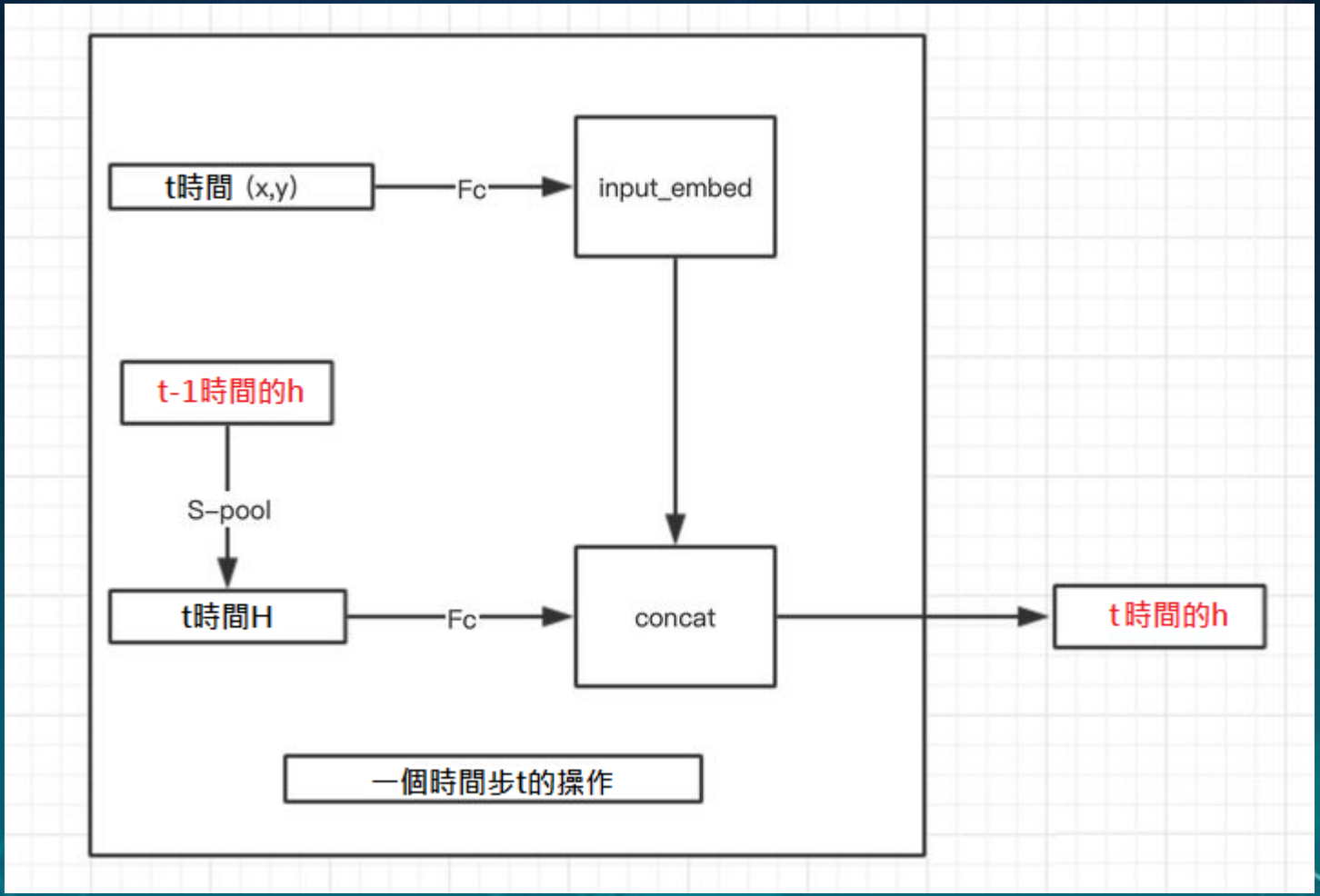


圖3. 一個step的資料流程程，將隱藏狀態來融合上一步中的鄰居資訊與當前位置資訊結合得到對應節點的當前隱藏狀態

位置估計

- 作者認為行人軌跡符合二維高斯分佈，預測得到的新的座標通過 $(\hat{x},\hat{y})_t^i \sim N(\mu_t^i, \sigma_t^i, \rho_t^i)$ 來進行預測,即通過隱藏狀態 h_{t-1}通過全連接層後利用**最大概似估計**(Maximum Likelihood Estimation, MLE)來逼近三個二維高斯函數的係數， 通過最小化概似估計來訓練資料集中的損失(loss)來訓練模型。
- 值得注意的是， 社交池化層(Social Pooling)沒有引入任何額外的參數。與傳統的LSTM的一個重要區別是，多個LSTM的隱藏狀態與社交池化層(Social Pooling)耦合，並且我們在每個時間步驟中都通過場景中的多個LSTM進行**聯合**反向傳播。

Occupancy map pooling

- 和S-pool基本類似，但這裡不使用上一層的隱藏層資訊作為鄰居節點資訊，而是使用當前節點周圍的位置資訊來進行融合。Social LSTM模型可以用來彙集來自相鄰的任何一組特徵。作為簡化，相關論文另外還實驗了一個模型，它只彙集鄰居的座標(稱為O-LSTM)。這是對原始模型的簡化，在訓練過程中不需要跨所有軌跡的聯合反向傳播。該模型仍然可以學習重新定位軌跡，以避免立即與鄰居發生碰撞。然而，在沒有來自鄰居的更多資訊的情況下，該模型將無法平滑的改變路徑以避免將來的碰撞。
- 具體公式：$O_t^i(,n,:) = \sum_{j \in N_i} 1mn\left[x_t^j - x_t^i, y_t^j - y_t^i\right]$
- 簡單的彙集了一下節點周圍每個區域的鄰居數量，但比起Social LSTM，缺少了前一次狀態的隱藏資訊。

論文的實驗結果

Metric	Methods	Lin	LTA	SF [73]	IGP* [60]	LSTM	our O-LSTM	our Social-LSTM
Avg. disp. error	ETH [49]	0.80	0.54	0.41	**0.20**	0.60	0.49	0.50
	HOTEL [49]	0.39	0.38	0.25	0.24	0.15	**0.09**	0.11
	ZARA 1 [39]	0.47	0.37	0.40	0.39	0.43	**0.22**	**0.22**
	ZARA 2 [39]	0.45	0.40	0.40	0.41	0.51	0.28	**0.25**
	UCY [39]	0.57	0.51	0.48	0.61	0.52	0.35	**0.27**
	Average	0.53	0.44	0.39	0.37	0.44	0.28	**0.27**
Avg. non-linear disp. error	ETH [49]	0.95	0.70	0.49	0.39	0.28	**0.24**	0.25
	HOTEL [49]	0.55	0.49	0.38	0.34	0.09	**0.06**	0.07
	ZARA 1 [39]	0.56	0.39	0.41	0.54	0.24	**0.13**	**0.13**
	ZARA 2 [39]	0.44	0.41	0.39	0.43	0.30	0.20	**0.16**
	UCY [39]	0.62	0.57	0.54	0.62	0.31	0.20	**0.16**
	Average	0.62	0.51	0.44	0.46	0.24	0.17	**0.15**
Final disp. error	ETH [49]	1.31	0.77	0.59	**0.43**	1.31	1.06	1.07
	HOTEL [49]	0.55	0.64	0.37	0.37	0.33	**0.20**	0.23
	ZARA 1 [39]	0.89	0.66	0.60	0.39	0.93	**0.46**	0.48
	ZARA 2 [39]	0.91	0.72	0.68	0.42	1.09	0.58	**0.50**
	UCY [39]	1.14	0.95	0.78	1.82	1.25	0.90	**0.77**
	Average	0.97	0.74	**0.60**	0.69	0.98	0.64	0.61

圖4. 結果展示，整體而言，論文所提出的方法效果主要是在密集場所更為有效，這也和方法所提出的收集附近節點思路相符合

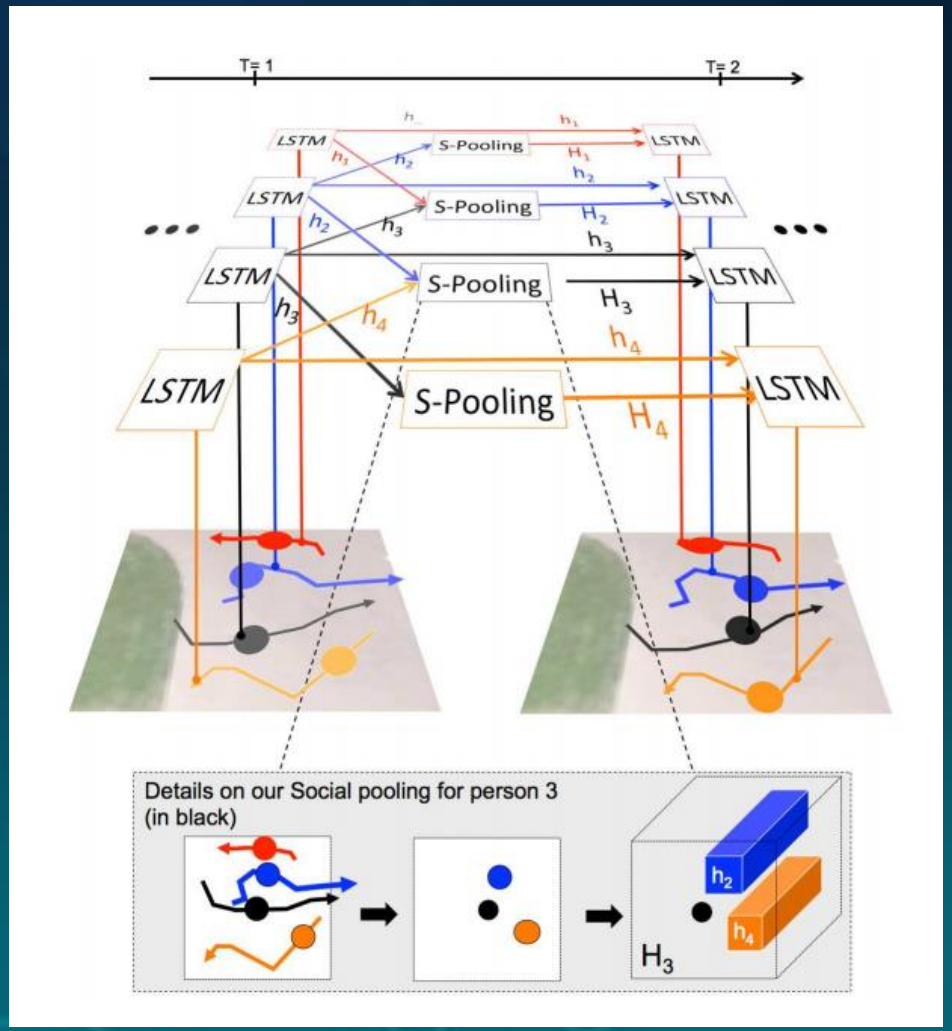
論文的實驗結果

- 從實驗結果表可以看出O-LSTM是對原始模型(Social LSTM)的簡化，在訓練過程中不需要跨所有軌跡的聯合反向傳播，該模型仍然可以學習重新定位軌跡，以避免立即與鄰居發生碰撞。然而，在沒有足夠多來自鄰居的資訊的情況下，該模型將無法平滑的改變路徑以避免未來的碰撞。

Social LSTM 軌跡預測專案

Social LSTM 軌跡預測

- 本專案是利用史丹佛大學研究團隊所錄製的資料集，進行行人軌跡預測。
- 架構流程：
  - 以LSTM作為Encoder將軌跡轉為content vector後，再送進Social Pooling進行處理
  - 再將Pooling後的結果用另一個LSTM作為Decoder來進行預測。

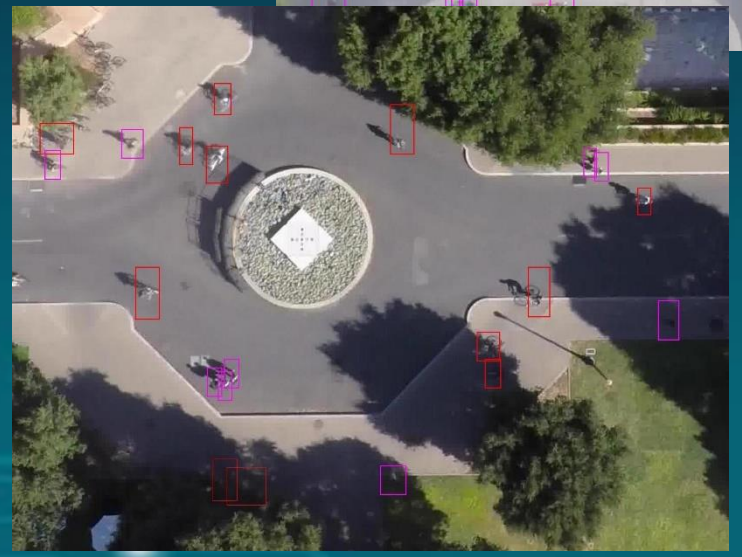


Social LSTM 軌跡預測

- 輸入：場景中所有交通物件的軌跡 $\mathbf{X} = X_1, X_2, \ldots, X_n$，其中 n 是交通物件的數量
- 輸出：預測出的軌跡 $\hat{\mathbf{Y}} = \hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_n$
- 對於每個交通物件 i，它的輸入軌跡以 $X_i = \left(x_i^t, y_i^t\right)$ 表示，其中 $t = 1, \ldots, t_{obs}$
- 而它的未來軌跡(ground truth)以 $Y_i = \left(x_i^t, y_i^t\right)$ 表示，其中 $t = t_{obs+1}, \cdots, t_{pred}$
- 預測的結果以 $\hat{Y}_i$ 表示

資料集介紹 – Stanford Drone Dataset

- 由史丹佛大學研究團隊所建立
- 錄製八個不同的校園場景，共包含有60部影片
- 影片中包含有自行車、行人、滑板、推車、汽車及巴士等類別
- 其中以自行車及行人類別最為充足，佔90%以上
- 是軌跡預測任務中常用來進行實驗的資料集



專案下載

- (本專案推薦使用Google Colab雲端計算平台，也可使用自己的電腦進行訓練)
- 首先打開終端機，下指令：
  git clone https://github.com/quancore/social-lstm

```
!git  clone  https://github.com/quancore/social-lstm
```

- 下載好後進入資料夾：
  cd ./social-lstm

```
%cd  social-lstm
```

訓練資料集調整

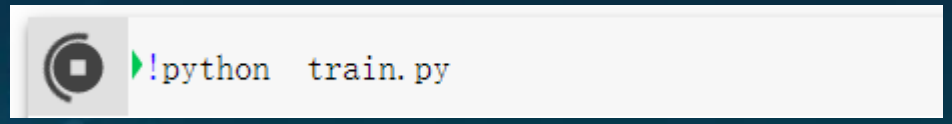
- 首先打開utils.py，為了使用Stanford資料集進行訓練，修改base_train_dataset中的註解

utils.py

```
22              forcePreProcess : Flag to forcefully preprocess the data again
23              '''
24              # base test files
25              base_test_dataset=    [#'/data/test/biwi/biwi_eth.txt',
26                                     #'/data/test/crowds/crowds_zara01.txt',
27                                     #'/data/test/crowds/uni_examples.txt',
28                                     #'/data/test/stanford/coupa_0.txt',
29                                      '/data/test/stanford/coupa_1.txt',  '/
30                                       #'/data/test/stanford/little_0.txt',
31                                       #'/data/test/stanford/quad_0.txt','/
32                                       ]
33              #base train files
34              base_train_dataset  =  [#'/data/train/biwi/biwi_hotel.txt',
35                                     #'/data/train/crowds/arxiepiskopi1.txt',
36                                     #'/data/train/crowds/crowds_zara03.txt',
37                                     #'/data/train/mot/PETS09-S2L1.txt',
38                                     '/data/train/stanford/bookstore_0.txt','
39                                     #'/data/train/stanford/deathCircle_4.txt
40                                     #'/data/train/stanford/hyang_5.txt','/da
41                                     ]
42              # dimensions of each file set
43              self.dataset_dimensions  =  {'biwi':[720,  576],  'crowds':[720,  576],
44
```

訓練模型

- 在終端機中輸入指令即可訓練模型:
  python3 train.py



訓練參數介紹

- --input_size:輸入的x、y座標
- --output_size:輸出的結果(效果不知)
- --rnn_size:隱藏層大小
- --batch_size:訓練時所用的封包大小(資料筆數)
- --seq_length: 輸入的觀測軌跡步數長度
- --pred_length: 輸出的預測軌跡步數長度

不可調整

可依據小組需求自行更改訓練參數以達到不同的訓練結果。

本次專案一律規定都需設為8

訓練參數介紹

- --num _epochs: 訓練的總次數
- --save_every: 每訓練幾次後儲存狀態
- --grad_clip: 控制梯度以免失控
- --learning_rate:每次訓練時調整參數的幅度
- --decay_rate: learning_rate的衰減率
- --embedding_size:嵌入參數的維度
- --neighborhood_size:決定鄰域大小的參數
- --grid_size:決定social 網格大小的參數

皆可依據小組需求自行更改訓練參數以達到不同的訓練結果。

訓練參數介紹

- --maxNumPeds:影響運算速率
- --lambda_param:對L2 loss function控制倍數
- --use_cuda: 使用GPU及CUDA函式庫進行訓練
- --gru:使用GRU還是LSTM架構
- --drive:是否使用Google drive
- --num_validation:驗證集數量
- --freq_validation:驗證頻率
- --freq_optimizer: optimizer的學習衰減頻率
- --grid:網格的儲存與後續使用(可能很佔記憶體)

皆可依據小組需求自行更改訓練參數以達到不同的訓練結果。

※加上訓練參數的範例：

```
!python train.py --num_epochs 5 --use_cuda
```

訓練過程

- 首先，程式會先對資料集進行讀取，並且針對軌跡數據進行前處理

```
allen@allen-All-Series:~/MyProject/social-lstm$ python3 train.py
Directory creation script is running...
Creating pre-processed validation data from raw data
Now processing:  ./data/validation/stanford/hyang_9_4.txt
Now processing:  ./data/validation/stanford/hyang_4_4.txt
Creating pre-processed training data from raw data
Now processing:  ./data/train/stanford/bookstore_0.txt
Now processing:  ./data/train/stanford/hyang_4.txt
Now processing:  ./data/train/stanford/hyang_6.txt
Now processing:  ./data/train/stanford/bookstore_0_0.txt
Now processing:  ./data/train/stanford/nexus_2.txt
Now processing:  ./data/train/stanford/hyang_6_0.txt
Now processing:  ./data/train/stanford/nexus_0.txt
Now processing:  ./data/train/stanford/nexus_4_2.txt
Now processing:  ./data/train/stanford/nexus_3.txt
Now processing:  ./data/train/stanford/gates_3_1.txt
Now processing:  ./data/train/stanford/gates_1.txt
```

- 讀取完後，程式會顯示出各資料集中的軌跡筆數

訓練過程(cont.)

- 接著開始進行訓練，程式會在每次訓練過程中顯示損失函數的值及每個封包(batch)執行的平均時間

```
****************Training epoch beginning******************
0/107280 (epoch 0), train_loss = 4.458, time/batch = 0.722
1/107280 (epoch 0), train_loss = 2.514, time/batch = 0.636
2/107280 (epoch 0), train_loss = 7.142, time/batch = 0.731
3/107280 (epoch 0), train_loss = 1.930, time/batch = 0.632
4/107280 (epoch 0), train_loss = 1.773, time/batch = 0.590
5/107280 (epoch 0), train_loss = 1.478, time/batch = 0.709
6/107280 (epoch 0), train_loss = 1.419, time/batch = 0.656
7/107280 (epoch 0), train_loss = 1.040, time/batch = 0.774
8/107280 (epoch 0), train_loss = 0.836, time/batch = 0.812
9/107280 (epoch 0), train_loss = 0.870, time/batch = 0.764
10/107280 (epoch 0), train_loss = 0.695, time/batch = 0.869
11/107280 (epoch 0), train_loss = 0.641, time/batch = 0.871
12/107280 (epoch 0), train_loss = 0.500, time/batch = 0.876
13/107280 (epoch 0), train_loss = 0.324, time/batch = 0.890
14/107280 (epoch 0), train_loss = 0.247, time/batch = 0.902
15/107280 (epoch 0), train_loss = 0.308, time/batch = 0.855
16/107280 (epoch 0), train_loss = 0.263, time/batch = 0.818
17/107280 (epoch 0), train_loss = 0.545, time/batch = 0.795
18/107280 (epoch 0), train_loss = 0.463, time/batch = 0.702
```

模型預測

- 在終端機中輸入指令即可進行預測:
  python3 test.py

```
!python  test.py

Selected method name:  SOCIALLSTM  model name:  LSTM
Creating pre-processed test data from raw data
Working on directory:  data/test/trajectories_test.cpkl
Now processing:  ./data/test/stanford/nexus_5.txt
Now processing:  ./data/test/stanford/hyang_3.txt
Now processing:  ./data/test/stanford/hyang_8.txt
Now processing:  ./data/test/stanford/little_2.txt
Now processing:  ./data/test/stanford/quad_2.txt
Now processing:  ./data/test/stanford/hyang_0.txt
Now processing:  ./data/test/stanford/little_3.txt
Now processing:  ./data/test/stanford/quad_0.txt
Now processing:  ./data/test/stanford/little_0.txt
Now processing:  ./data/test/stanford/quad_3.txt
Now processing:  ./data/test/stanford/nexus_6.txt
Now processing:  ./data/test/stanford/coupa_1.txt
Now processing:  ./data/test/stanford/coupa_0.txt
Now processing:  ./data/test/stanford/quad_1.txt
Now processing:  ./data/test/stanford/gates_2.txt
Now processing:  ./data/test/stanford/little_1.txt
Now processing:  ./data/test/stanford/hyang_1.txt
Loading train or test dataset:  data/test/trajectories_test.cpkl
Sequence size(frame) ------> 20
One batch size (frame)--->- 20          ...
```

模型驗證

- 在終端機中輸入指令即可進行驗證、獲得loss、mean_error等值:
  python3 validation.py

```
!python  validation.py
Current file :  hyang_6_3.txt  Batch :  1535  Sequence:  1
Current file :  hyang_6_3.txt  Batch :  1536  Sequence:  1
Current file :  hyang_6_3.txt  Batch :  1537  Sequence:  1
Current file :  hyang_6_3.txt  Batch :  1538  Sequence:  1
Current file :  hyang_6_3.txt  Batch :  1539  Sequence:  1
Current file :  hyang_6_3.txt  Batch :  1540  Sequence:  1
```

⋮

```
valid_loss = 7.701, valid_mean_err = 2.988, valid_final_err = 4.508
Writing to plot file  path: ./plot/SOCIALLSTM/LSTM/validation, file_name: bookstore_2_4.
Writing to plot file  path: ./plot/SOCIALLSTM/LSTM/validation, file_name: deathCircle_3_
Writing to plot file  path: ./plot/SOCIALLSTM/LSTM/validation, file_name: nexus_8_3.pkl
Writing to plot file  path: ./plot/SOCIALLSTM/LSTM/validation, file_name: gates_5_1.pkl
Writing to plot file  path: ./plot/SOCIALLSTM/LSTM/validation, file_name: hyang_6_3.pkl
```

...

模型誤差計算

在軌跡預測任務中，常用來驗證模型誤差的誤差測量有以下兩種：

- 平均位移誤差(Average Displacement Error, ADE)：
  以每個預測點與真實點的平均距離做為整條軌跡的平均誤差
- 終點位移誤差(Final Displacement Error, FDE)：
  只以最後一個預測點與真實點的距離作為整條軌跡的最後誤差

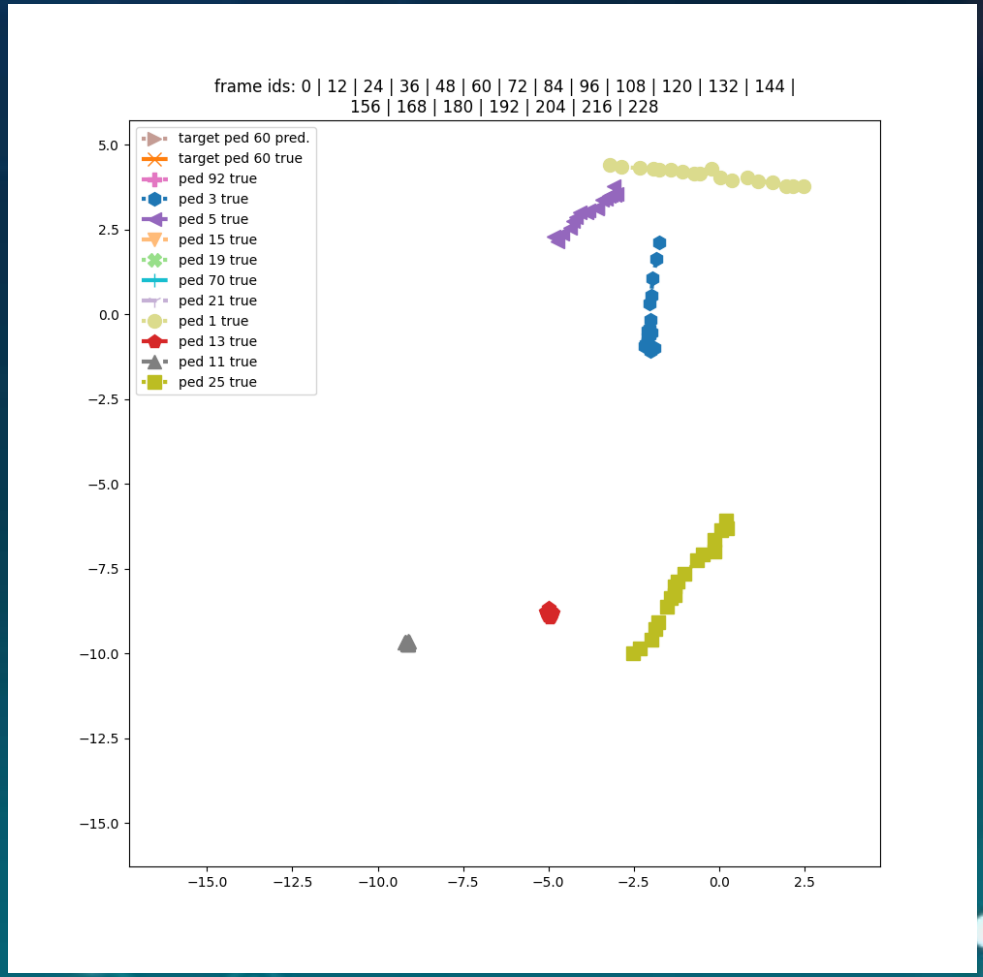
模型可視化結果

- 執行指令來產生模型可視化的結果：
  python3 visualize.py

```
!python  visualize.py
        [3.4186, 6.9578],
        [2.7373, 5.6077],
        [3.4444, 6.2938]])
Max number of peds in this  seq.:  15
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

target_id
[-2.4734921  -2.2735953  -2.2660637  -2.0383558  -1.8512077  -1.8720665
 -1.7283735  -1.7011414  -1.6843376  -1.6327705  -1.614809   -1.5806236
 -1.507618   -1.1321621  -0.8430376  -0.52262497 -0.24624729 -0.02491379]
 4.719549  4.735074  4.710883  4.714134  4.777318  4.816312  5.023197
 5.300851  5.625078  5.892621  6.272814 ]
[-2.4734921  -2.1470523  -2.1473384  -1.9460726  -2.0686703  -1.8880224
 -1.6601439  -1.5814772  -1.7011681  -1.2238388  -1.5351772  -0.9619503
 -1.487258   -1.3404064  -1.1158648  -0.89967823 -0.492733   -0.8874712 ]
 4.9670753 4.9610567 4.9747257 4.831151  5.4023066 5.2622166 5.170229
 5.367031  5.3801517 6.2536087 5.7661448]
Ped processing is aborted because its trajectory lenght in this sequence i
Ped processing is aborted because its trajectory lenght in this sequence i
Ped processing is aborted because its trajectory lenght in this sequence i
Ped processing is aborted because its trajectory lenght in this sequence i
Video creation for  sequence00087  is starting...
Video creation ended.
##################################################################################
Now processing seq:  sequence00088
****************************
orig true_
```

...



備註：由於資料集僅提供行人的座標，並無真實場景影片，因此可視化結果是一張plot在空白底圖上的xy座標圖，

參考資料

- 論文網址：
  - https://openaccess.thecvf.com/content_cvpr_2016/papers/Alahi_Social_LSTM_Human_CVPR_2016_paper.pdf

- Social LSTM論文筆記、Encoder-Decoder框架：
  - https://zhuanlan.zhihu.com/p/342149300?fbclid=IwAR1L_eDWSSqe5pp5n7ip_8BjDlhFIbROlnv18W1w1TmPu509unIN4tkxpSY
  - https://zhuanlan.zhihu.com/p/63396070?fbclid=IwAR1-bFN7-XJoxLdcM7gAMGAEs7xGpIyu0QWyQVasIOtFhQOjeslUeVPads0
  - https://iter01.com/558290.html?fbclid=IwAR1SH1l1DgXsGjXLaaUN38mqpFZV--xMRQnrQ80FxZyDFBH8iMQbanEnliQ